dot product cross product change of basis

April 20, 2019

In [1]:
```
%matplotlib inline
from ipywidgets import interactive
import matplotlib.pyplot as plt
import numpy as np



def unit_vector_xsys(theta, origin_reflection=False):

    v_hat_tip = np.array([np.cos(theta),np.sin(theta)])
    v_hat_tail = np.zeros(shape=(2,))

    if origin_reflection:
        v_hat_tail[0] = -v_hat_tip[0]
        v_hat_tail[1] = -v_hat_tip[1]


    v_xs = np.array([v_hat_tail[0], v_hat_tip[0]])
    v_ys = np.array([v_hat_tail[1], v_hat_tip[1]])

    return v_xs,v_ys


def plot_unit_vector_dot(theta1=0, theta2=1, theta_ihat=0):

    #
    # crete figure
    #

    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)

    #
    #  unit circle
    #

    # create unit circle
```

```
circ = plt.Circle((0, 0), radius=1, edgecolor='k', facecolor='None')

#
#  i hat
#
ihat_xs, ihat_ys = unit_vector_xsys(theta_ihat, origin_reflection=True)

#
#  j hat (orthogonal and ahead pi/2) [TMP]
#
theta_jhat = theta_ihat + np.pi/2
jhat_xs, jhat_ys = unit_vector_xsys(theta_jhat, origin_reflection=True)

#
# vector 1
#

v1_xs,v1_ys = unit_vector_xsys(theta1, origin_reflection=False)

#
# vector 1 i projection)
#

v1_proi_xs = np.array([ihat_xs[1]*v1_xs[1],0])
v1_proi_ys = np.zeros(shape=(2,))

#
# vector 1 j projection)
#

v1_proj_xs = np.zeros(shape=(2,))
v1_proj_ys = np.array([jhat_ys[1]*v1_ys[1],0])


#
# vector 2
#

v2_xs,v2_ys = unit_vector_xsys(theta2, origin_reflection=False)

#
# vector 2 counter part (+pi/2)
#
v2_counter_theta = theta2 +np.pi/2
v2_counter_xs,v2_counter_ys = unit_vector_xsys(v2_counter_theta, origin_reflection=

#
# vector 2 counter j projection (vector 2 i projection)
```

```
#

### catastrophic cancelation
#counter_proj_j_theta = np.dot(v2_counter_ys,jhat_ys) #np.cos(v2_counter_theta)
#counter_proj_xs, counter_proj_ys = unit_vector_xsys(counter_proj_j_theta, origin_
### catastrophic cancelation

counter_proj_xs = np.zeros(shape=(2,))
counter_proj_ys = np.array([0, jhat_ys[1]*v2_counter_ys[1]])


#
# vector 2 counter i projection (vector 2 j projection)
#
counter_proi_xs = np.array([ihat_xs[1]*v2_counter_xs[1],0])
counter_proi_ys = np.zeros(shape=(2,))


#
#
#

#
# Plot Things
#

# plot i hat
ax.plot(ihat_xs, ihat_ys, 'k--', linewidth=0.5)

# plot j hat
ax.plot(jhat_xs, jhat_ys, 'k--', linewidth=0.5)

# plot vector 1
#ax.plot(v1_xs, v1_ys, c='b')
plt.quiver(v1_xs[0],v1_ys[0], v1_xs[1], v1_ys[1], color='b', scale=4)

# plot vector 1 projection onto i hat
ax.plot(v1_proi_xs, v1_proi_ys, 'b', linewidth=1)

# plot vector 1 projection onto j hat
ax.plot(v1_proj_xs, v1_proj_ys, 'b', linewidth=1)



# plot vector 2
#ax.plot(v2_xs, v2_ys, c='r')
plt.quiver(v2_xs[0],v2_ys[0], v2_xs[1], v2_ys[1], color='r', scale=4)
```

```
            # plot vector 2 counter part
            ax.plot(v2_counter_xs,v2_counter_ys, 'r--', linewidth=0.5, alpha=0.25)

            # # plot vector 2 counter part projection onto j hat
            ax.plot(counter_proj_xs, counter_proj_ys, c='r', linewidth=1)

            # # plot vector 2 counter part projection onto i hat
            ax.plot(counter_proi_xs, counter_proi_ys, c='r', linewidth=1)

            # plot circle
            ax.add_patch(circ)

            # plot origin
            ax.plot(0,0,marker='o', c='black')

            #
            # Plot settings
            #

            # set equal aspect
            ax.set_aspect('equal', 'datalim')

            # plot bounds
            plt.ylim(-2, 2)
            plt.xlim(-2, 2)

            # add grid
            plt.grid(False)

            # display
            plt.show()

        interactive_plot = interactive(plot_unit_vector_dot, theta1=(0.0, 2*np.pi, 0.25), theta
        output = interactive_plot.children[-1]
        output.layout.height = '350px'
        interactive_plot
```

```
interactive(children=(FloatSlider(value=0.0, description='theta1', max=6.283185307179586, step=
```